BD

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-265612

(43)Date of publication of application : 28.09.2001

(51)Int.Cl. G06F 9/46

G06F 9/455

G06F 13/14

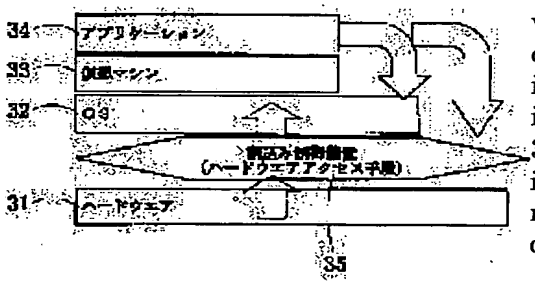
(21)Application number : 2000-076568

(22)Date of filing : 17.03.2000

(71)Applicant : OMRON CORP

(72)Inventor : HIRONO MITSUAKI
KONAKA YOSHIHARU
KADOWAKI MASANORI
MURAI KENICHI
SAIKI HIDEAKI

(54) VIRTUAL MACHINE SYSTEM FOR INCORPORATED EQUIPMENT



(57)Abstract:
PROBLEM TO BE SOLVED: To directly execute a hardware access processing from an application without using a device driver.
SOLUTION: An interruption controller 35 is provided between hardware 31 and an operating system 32 and a virtual machine 33 and the application 34 are constituted on the operating system 32. When interruption is present from the hardware 31, an interruption processor managed by the operating system 32 is activated, the application 34 activates the interruption controller 35 by API bypassing the virtual machine 33 in response to that and the hardware 31 is directly controlled.

LEGAL STATUS

[Date of request for examination] 27.11.2003

[Date of sending the examiner's decision of 13.06.2006

rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

CLAIMS

[Claim(s)]
[Claim 1] The virtual machine system for incorporated appliances equipped with the 1st hardware access means which is constituted by application, a virtual machine, and the operating system and is managed by the operating system, and the 2nd hardware access means managed by application or the virtual machine.
[Claim 2] Said 2nd hardware access means is a virtual machine system for incorporated appliances according to claim 1 characterized by being what performs hardware access processing in relation to said 1st hardware access means.
[Claim 3] Said each of 1st and 2nd hardware access means are the virtual machine systems for incorporated appliances according to claim 1 which were an interrupt controller, and could perform interrupt processing with application or a virtual machine while it was connected with the interrupt handler of an operating system.
[Claim 4] The virtual machine system for incorporated appliances according to claim 1 characterized by said virtual machine being a Java (trademark) virtual machine.

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Field of the Invention] Especially this invention relates to the virtual machine system for incorporated appliances in a Java execution environment about the virtual machine system for incorporated appliances.
[0002]
[Background of the Invention] In the conventional computer system, as shown in drawing 1 , an operating system (OS) 2 is placed on the configuration of hardware 1, and the various applications 3 are moved by making an operating system 2 into a platform. Although a computer system has the architecture of the proper depending on the configuration of hardware 1, an operating system 2 absorbs the difference in hardware 1, and enables it to operate application 3 on a different hardware configuration. Moreover, what changed the application 3 for activation into the machine language from the source code 4 using the assembler, the compiler, etc. is used.
[0003] However, it depended on the operating system 2 for application 3 (what is depended on a machine language) at such a computer system, and since it was necessary to compile to a machine language which is different if operating systems 2 differ even if a source code 4 is the

same, application 3 was not able to be moved on a different operating system 2 (for example, Windows (trademark), Unix (trademark), Linux, etc.).
[0004] Then, as shown in drawing 2 , there is a design that I will make it the application 14 which was made to absorb the difference in hardware 11 or an operating system 12 with a virtual machine 13, and was compiled with Java, a C++ compiler, etc. move on any hardware 11 and operating systems 12, conventionally by building a virtual machine 13 on an operating system 12. A virtual machine (virtual machine;VM) 13 is the computer (software) with logical architecture formed on the computer, and an operating system and hardware are supposed and it enables it to treat with a common interface (I/F) here. Especially, the Java virtual machine of Sun Microsystems, Inc. (Sun Microsystems) attracts attention in recent years. Java2 Platform and Micro Edition are announced especially as an object for incorporated appliances.
[0005] Moreover, in the object for incorporated appliances, and the other software developments of **, in order to raise the productivity, it is thought that object-oriented languages, such as C++ and Java, become in use from now on. Especially, according to Java language, in the environment where the Java virtual machine is mounted, the code (cutting tool code) independent of hardware is generable.
[0006] However, in for incorporated appliances, as shown in drawing 3 , ROM17 and the memory of RAM18 grade are connected with CPU (microprocessor)16 through the bus line 20, and interruption from the various peripheral devices 19 may enter through the lines 21, such as a LAN controller, and a serial boat, a parallel port. Therefore, in activation of application, the approach of controlling these interruption poses a problem.
[0007] If in the case of the computer system which does not mount the virtual machine the interrupt by the asynchronous access signal occurs as are shown in drawing 4 , and application 3 is described to operating system 2 and hardware 1 dedication and it is shown in drawing 5 (step S1), the I/O access instruction of a microprocessor is executed (step S2), and it has become the structure which controls direct hardware (peripheral device).
[0008] However, as shown in drawing 6 , in order to give the similarity as language, direct access to an operating system 12 or hardware 11 from application 14 is not allowed with the system which mounted the virtual machine expected to become the mainstream from now on, for example, a Java execution environment. Therefore, it is impossible to incorporate directly by the high level language or Java, and to create the application 14 of a device, and it is required to solve this.
[0009] Therefore, in the case of the system which mounted the virtual machine 13, the device driver 22 for controlling a peripheral device to be shown in drawing 7 is described by the assembler or C, and application 14 serves as structure which accesses hardware 11 by the device driver 22 through the common interface (I/F) 23.
[0010] With such a configuration, in order to access hardware 11 If the interrupt request of the asynchronous access signal is outputted and carried out as shown in the flow of drawing 8 (step S11) A device driver 22 is opened by API (Application Programing Interface) of a virtual machine 13 by accessing a virtual machine 13 through the common interface 23 (step S12). Subsequently, access an operating system 12 through the common interface 23, and a device driver 22 is opened by API of an operating system 12 (step S13). Furthermore, a hardware access request is outputted by API of a virtual machine 13 (step S14), and a hardware access request is outputted by API of an operating system 12 (step S15). By this, a device driver 22 accesses hardware 11 and a device driver (step S16) 22 returns a result (step S17). Then, application 14 needs processing in which close a device driver 22 by API of a virtual machine 13 (step S18), and a device driver 22 is closed by API of an operating system 12 (step S19).
[0011] Or as shown in drawing 9 , the common device driver 24 for controlling a peripheral device is formed, and application 14 may access hardware 11 by the common device driver 24 through the common interface (I/F) 23.
[0012] However, by these approaches, apart from application, since the device driver (common) had to be described by the assembler or C, since the fabrication of a device driver was needed, development effectiveness was bad, and it or before and productivity did not change. Moreover, control of fine hardware was not completed by the approach of accessing hardware through a common interface. Furthermore, since API was called by multiplex, there was a problem that processing speed was slow.
[0013] Moreover, it is important that hardware 11 is finely controllable by the incorporated

appliance from application 14 to the similarity and coincidence of language. Therefore, in the latest measure, API of the bypass which can carry out direct access is prepared for an operating system or hardware, and they are supported. However, the operating system 12 was carrying out unitary management, and the resource important for actuation of the operating systems 12, such as interruption, had it in the system which mounted the virtual machine 13. [impossible for bypassing a virtual machine 13 like drawing 10]

[0014] Next, in the conventional interruption processor, it interrupted and interruption processing was performed [prepared / a controller 26 or interrupting, while receiving interruption from a peripheral device, and supervising / having been prepared in the exterior of CPU16, as shown in drawing 12 / the condition of the source by the controller 26, by interrupting, / in the interior of CPU16 / as shown in drawing 11].

[0015] Drawing 13 expresses the interrupt-processing equipment which interrupted the interior of CPU16 and formed the controller 28. If an external interrupt request is permitted, the interrupt from the interruption source 27 will occur (C1), and the interruption controller 28 will receive interruption. and -- if interruption is received -- an operating system 12 -- interrupting -- blocking -- others (C2) -- interruption is forbidden. Subsequently, the OS interrupt handler 29 is started (C3), and a semaphore notifies rising to the Java virtual machine interrupt handler 30 (C4). If the Java virtual machine interrupt handler 30 rises, the interruption processing by Java virtual machine 13 will occur (C5). The started OS interrupt handler 29 supervises progress of interruption processing by communicating with the interruption controller 28 (C6). And if the OS interrupt handler 29 interrupts, the interruption condition of the source 27 is canceled (C7), an operating system 12 interrupts, after interruption processing is completed, and a block is canceled (C8), the OS interrupt handler 29 will escape from a handler, and it will come out of it (C9).

[0016] Thus, package management is carried out by the operating system, and interruption processing was able to be controlled by the conventional interruption processor neither with application nor a virtual machine. For this reason, the effort which must perform by interrupting by controlling hardware by the device driver as mentioned above, and creates a device driver with creation of application was required.

[0017]

[Description of the Invention] The place made into the object of this invention is to enable it to perform direct hardware access processing from application, without using a device driver.

[0018] The virtual machine system for incorporated appliances concerning this invention is constituted by application, a virtual machine, and the operating system, and is equipped with the 1st hardware access means managed by the operating system, and the 2nd hardware access means managed by application or the virtual machine.

[0019] Since the virtual machine system for incorporated appliances concerning this invention is equipped with the hardware access means managed by application or the virtual machine apart from the hardware access means managed by the operating system, it can perform hardware access processing also with application or a virtual machine. Therefore, there is no need of writing a device driver to accessing hardware from application or a virtual machine, and the program for accessing hardware can be written in application. Therefore, development of application can be made easy.

[0020] In the operation gestalt of this invention, said 2nd hardware access means performs hardware access processing in relation to said 1st hardware access means. Since the hardware access means managed by application or the virtual machine is connected with the hardware access means managed by the operating system, after an operating system starts access to hardware, it enables application or a virtual machine to succeed from an operating system, and to continue and access hardware. Therefore, application and a virtual machine can access hardware now also by the system which can access hardware directly neither from application nor a virtual machine.

[0021] Moreover, in another operation gestalt of this invention, said each of 1st and 2nd hardware access means are interrupt controllers, and they can perform interrupt processing with application or a virtual machine, being connected with the interrupt handler of an operating system. After an operating system receives an interrupt especially in the case of interrupt processing, interrupt processing can be passed to application or a virtual machine with an interrupt handler.

[0022] Especially as a virtual machine, the Java virtual machine attracts attention and a response becomes easy also as an object for incorporated appliances.
[0023] In addition, the component explained the above of this invention is combinable as much as possible.
[0024]
[Embodiment of the Invention] (1st operation gestalt) The virtual machine system for incorporated appliances by 1 operation gestalt of this invention is shown in drawing 14 . If it is in this virtual machine system for incorporated appliances, interrupt control equipment (hardware access means) 35 is formed between hardware 31 and an operating system 32, and application 34 is performed on Java virtual machine 33 mounted on the operating system 32.
[0025] If it is in this virtual machine system for inclusion devices, the interrupt request from hardware 31 is inputted into an operating system 32 through interrupt control equipment 35. If an interrupt request is received, an operating system 32 will perform interrupt processing. Application 34 accesses interrupt control equipment 35 in cooperation in relation to an operating system 32 interrupting and processing, and performs interruption processing with interrupt control equipment 35.
[0026] As an approach application 34 accesses interrupt control equipment 35, also by API which bypasses a virtual machine 33, it is good, and from a virtual machine 33, an operating system 32 may be bypassed and may be accessed.
[0027] A deer is carried out, according to this system, in harmony with an operating system 32, direct interrupt control equipment 35 is accessed from application 34, and an interrupt can be controlled.
[0028] Thus, since interrupt control equipment 35 is formed between an operating system 32 and hardware 31 and it enables it to control interrupt control equipment 35 from application 34, it is not necessary to write a device driver like before, the program for interruption processing can be written into application 34, and the productivity of application 34 can be raised. Moreover, since API is not started by multiplex like before, processing speed improves.
[0029] As an approach application 34 accesses hardware, a virtual machine 33 may be bypassed, for example, interrupt control equipment 35 may be accessed, and interrupt control equipment 35 may be accessed via a virtual machine 33. Drawing 15 shows how to control interrupt control equipment 35 by API which bypasses a virtual machine 33. By this approach, if there is an access request with an asynchronous access signal first as shown in the flow of drawing 15 (step S21), the I/O object which accesses interrupt control equipment 35 by API which bypasses a virtual machine 33 will be created (step S22), and hardware 31 will be accessed using this I/O object (step S23). Then, the I/O object which accesses interrupt control equipment 35 by API which bypasses a virtual machine 33 is deleted (step S24).
[0030] Moreover, drawing 16 expresses the another approach of accessing hardware. That is, if there is an access request with an asynchronous access signal (step S31), the I/O object which accesses interrupt control equipment 35 by API which bypasses a virtual machine 33 will be created (step S32), and an access request will be carried out to a hardware access means (interrupt control equipment 35) using the I/O object (step S33). A hardware access means restricts the function of an operating system 32, or changes the actuation (step S34), and accesses hardware 31 using an I/O object (step S35). And if interruption processing is completed, a hardware access means (interrupt control equipment 35) will restrict the function of an operating system 32, or will cancel modification of the actuation (step S36), and will delete an I/O object by API which subsequently bypasses a virtual machine 33 (step S37).
[0031] Drawing 17 shows the structure for forming the external interruption controller (interrupt control equipment) 35 between hardware 31 and an operating system 32, and operating interruption processing from application 34. this interruption art -- an operating system 32 -- an external interrupt request -- receiving -- interruption -- granting a permission (C11) -- an interrupt occurs from the interruption source 37 (C12), the external interruption controller 35 receives interruption, and interruption is inputted into the interrupt controller 38 of the CPU36 interior from the external interruption controller 35 (C13). and -- if interruption is received -- CPU36 -- an operating system 12 -- interrupting -- blocking -- others (C14) -- interruption is forbidden. Subsequently, an integrated interrupt handler (OS level) is started (C15). The integrated interrupt handler 39 makes disapproval external interruption of an additional interruption block function (C16), a semaphore notifies rising to the Java virtual

machine interrupt handler 40 (C17), the integrated interrupt handler 39 exits from the loop formation of interruption processing (C18), and an operating system 32 cancels the interruption block in an interrupt controller 38 (C19). On the other hand, the java virtual machine interrupt handler 40 rises, performs interruption processing by Java virtual machine 33 (C20), and supervises interrupt processing by communicating with the external interruption controller 35 (or interruption source 37) (C21). After interrupt processing by Java virtual machine 33 is completed, the Java virtual machine interrupt handler 40 ends interrupt processing (C22), and the integrated interrupt handler 39 carries out external interruption to authorization in an interrupt controller 38 (C23).

[0032] In addition, in drawing 17 , although the interruption controller is formed in the interior of CPU, it interrupts with an external interruption controller and the controller may also be formed in the exterior of CPU.

[0033]

[Effect of the Invention] According to the virtual machine system for incorporated appliances of this invention, the hardware access means has doubled, and since it operates in relation to the hardware access means by the side of an operating system, the 2nd hardware access means managed by application etc. can perform direct hardware access processing now from application, and can aim at the productivity drive of application, and improvement in processing speed.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]

[Drawing 1] It is drawing showing the configuration of the conventional general computer system.

[Drawing 2] It is drawing showing the configuration of the conventional computer system equipped with the virtual machine.

[Drawing 3] It is drawing showing the relation between a microprocessor and a peripheral device.

[Drawing 4] In the computer system of drawing 1 , it is drawing showing signs that hardware is controlled directly from application.

[Drawing 5] In a computer system same as the above, it is flow drawing explaining the procedure which accesses hardware.

[Drawing 6] In the computer system equipped with the virtual machine, it is drawing showing signs that it cannot access from application to an operating system or hardware.

[Drawing 7] It is drawing showing the computer system it enabled it to access from application to hardware using a device driver and a common interface.

[Drawing 8] It is drawing explaining the procedure accessed from application to hardware by the approach same as the above.

[Drawing 9] It is drawing showing the computer system it enabled it to access from application to hardware using a common device driver and a common interface.

[Drawing 10] It is drawing showing the computer system it enabled it to access from application to hardware using API which bypasses a virtual machine.

[Drawing 11] It is drawing showing the conventional system which formed the interrupt controller in the interior of a microprocessor.

[Drawing 12] It is drawing showing the conventional system which formed the interrupt controller in the exterior of a microprocessor.

[Drawing 13] It is the block diagram showing the configuration of conventional interrupt-processing equipment.

[Drawing 14] It is the schematic diagram showing the virtual machine structure of a system for incorporated appliances by 1 operation gestalt of this invention.

[Drawing 15] In the virtual machine for incorporated appliances same as the above, it is flow drawing explaining how to access hardware.

[Drawing 16] In the virtual system for incorporated appliances of drawing 14 , it is flow drawing explaining the another approach of accessing hardware.

[Drawing 17] It is drawing showing the configuration of the interrupt-processing equipment

concerning this invention.
[Description of Notations]
31 Hardware
32 Operating System
33 Virtual Machine
34 Application
35 Control Unit

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2001－265612
（P2001－265612A）
(43)公開日　平成13年９月28日(2001.9.28)

(51)Int.Cl.⁷	識別記号	ＦＩ	テーマコード（参考）
Ｇ０６Ｆ　9/46	３４０	Ｇ０６Ｆ　9/46	３４０Ｆ　５Ｂ０１４
9/455		13/14	３１０Ｊ　５Ｂ０９８
13/14	３１０	9/44	３１０Ａ

審査請求　未請求　請求項の数４　ＯＬ　（全 11 頁）

(21)出願番号　特願2000－76568(P2000－76568)

(22)出願日　平成12年３月17日(2000.3.17)

(71)出願人　000002945
オムロン株式会社
京都市下京区塩小路通堀川東入南不動堂町801番地

(72)発明者　廣野　光明
京都府京都市右京区花園土堂町10番地　オムロン株式会社内

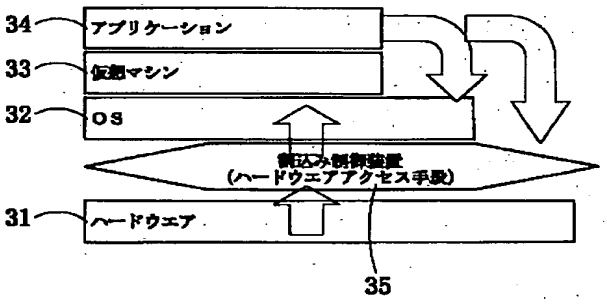
(72)発明者　小中　義治
京都府京都市右京区花園土堂町10番地　オムロン株式会社内

(74)代理人　100094019
弁理士　中野　雅房

最終頁に続く

(54)【発明の名称】　組込み機器用仮想マシンシステム

(57)【要約】
【課題】　デバイスドライバを用いることなく、アプリケーションから直接ハードウエアアクセス処理を実行できるようにする。
【解決手段】　ハードウエア３１とオペレーティングシステム３２との間に割込み制御装置３５を設け、オペレーティングシステム３２の上に仮想マシン３３とアプリケーション３４を構成する。ハードウエア３１から割込みがあると、オペレーティングシステム３２によって管理される割り込み処理装置が起動し、それに応答して、仮想マシン３３をバイパスするＡＰＩによってアプリケーション３４が割込み制御装置３５を起動し、ハードウエア３１を直接に制御する。



1

【特許請求の範囲】

【請求項１】　アプリケーションと、仮想マシンと、オペレーティングシステムとによって構成され、

オペレーティングシステムによって管理される第１のハードウエアアクセス手段と、アプリケーション又は仮想マシンによって管理される第２のハードウエアアクセス手段とを備えた組込み機器用仮想マシンシステム。

【請求項２】　前記第２のハードウエアアクセス手段は、前記第１のハードウエアアクセス手段と関連してハードウエアアクセス処理を実行するものであることを特徴とする、請求項１に記載の組込み機器用仮想マシンシステム。

【請求項３】　前記第１及び第２のハードウエアアクセス手段は、いずれも割込みコントローラであって、オペレーティングシステムの割込ハンドラと関連しながらアプリケーション又は仮想マシンにより割込み処理を実行できるようになった、請求項１に記載の組込み機器用仮想マシンシステム。

【請求項４】　前記仮想マシンがＪａｖａ（登録商標）仮想マシンであることを特徴とする、請求項１に記載の組込み機器用仮想マシンシステム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、組込み機器用仮想マシンシステムに関し、特にＪａｖａ実行環境における組込み機器用仮想マシンシステムに関する。

【０００２】

【背景技術】従来のコンピュータシステムにおいては、図１に示すように、ハードウエア１の構成上にオペレーティングシステム（ＯＳ）２を置き、オペレーティングシステム２をプラットホームとして種々のアプリケーション３を動かしている。コンピュータシステムはハードウエア１の構成に依存した固有のアーキテクチャを持つが、オペレーティングシステム２は、ハードウエア１の違いを吸収し、異なるハードウエア構成の上でアプリケーション３が動作できるようにしている。また、実行用のアプリケーション３は、アセンブラやコンパイラなどを用いてソースコード４からマシン語に変換したものが用いられる。

【０００３】しかし、このようなコンピュータシステムでは、アプリケーション３（マシン語によるもの）はオペレーティングシステム２に依存しており、ソースコード４が同じであってもオペレーティングシステム２が異なれば異なるマシン語にコンパイルする必要があるため、アプリケーション３は異なるオペレーティングシステム２（例えば、Ｗｉｎｄｏｗｓ（登録商標）、Ｕｎｉｘ（登録商標）、Linuxなど）の上では動かすことができなかった。

【０００４】そこで、図２に示すように、オペレーティングシステム１２の上に仮想マシン１３を構築することにより、仮想マシン１３でハードウエア１１やオペレーティングシステム１２の違いを吸収させ、ＪａｖａやＣ++コンパイラ等でコンパイルされたアプリケーション１４がどのようなハードウエア１１やオペレーティングシステム１２の上でも動くようにしようという構想が従来よりある。ここで、仮想マシン（virtual machine；ＶＭ）１３とは、コンピュータ上に形成された、論理的なアーキテクチャを持つコンピュータ（ソフトウエア）であって、オペレーティングシステムやハードウエアを仮想化して共通のインターフェイス（Ｉ／Ｆ）で扱えるようにするものである。なかでも、近年サン・マイクロシステムズ（Sun Microsystems）社のＪａｖａ仮想マシンが注目されている。特に、組込み機器用としては、Java２ Platform, Micro Editionが発表されている。

【０００５】また、組込み機器用その他ののソフトウエア開発では、その生産性を向上させるため、今後はＣ++やＪａｖａ等のオブジェクト指向言語が主流になっていくものと思われる。特に、Ｊａｖａ言語によれば、Ｊａｖａ仮想マシンが実装されている環境では、ハードウエアに依存しないコード（バイトコード）を生成することができる。

【０００６】しかし、組込み機器用の場合には、ＣＰＵ（マイクロプロセッサ）１６には、図３に示すように、バスライン２０を通じてＲＯＭ１７やＲＡＭ１８等のメモリがつながれており、またＬＡＮコントローラやシリアルポート、パラレルポートなどのライン２１を通じて各種周辺機器１９からの割り込みが入ることがある。従って、アプリケーションの実行においては、これらの割り込みを制御する方法が問題となる。

【０００７】仮想マシンを実装していないコンピュータシステムの場合には、図４に示すように、オペレーティングシステム２やハードウエア１専用にアプリケーション３を記述しておき、図５に示すように、非同期アクセス信号による割り込みが発生すると（ステップＳ１）、マイクロプロセッサのＩ／Ｏアクセス命令を実行し（ステップＳ２）、直接ハードウエア（周辺機器）を制御する仕組みとなっている。

【０００８】しかし、今後主流になっていくと思われる仮想マシンを実装したシステム、例えばＪａｖａ実行環境では、図６に示すように、言語としての共通性を持たせるために、アプリケーション１４からオペレーティングシステム１２やハードウエア１１への直接のアクセスを許していない。従って、高級言語やＪａｖａで直接に組み込み機器のアプリケーション１４を作成することは不可能であり、これを解決することが必要である。

【０００９】そのため、仮想マシン１３を実装したシステムの場合には、図７に示すように、周辺機器を制御するためのデバイスドライバ２２をアセンブラやＣ言語で記述しておき、アプリケーション１４は共通インターフェイス（Ｉ／Ｆ）２３を介してデバイスドライバ２２で

ハードウエア１１にアクセスする仕組みとなっている。

【００１０】このような構成では、ハードウエア１１にアクセスするためには、図８のフローに示すように、非同期アクセス信号を出力して割込み要求すると（ステップＳ１１）、共通インターフェイス２３を介して仮想マシン１３にアクセスすることにより仮想マシン１３のＡＰＩ（Application Programing Interface）でデバイスドライバ２２をオープンし（ステップＳ１２）、ついで共通インターフェイス２３を介してオペレーティングシステム１２にアクセスしてオペレーティングシステム１２のＡＰＩでデバイスドライバ２２をオープンし（ステップＳ１３）、さらに、仮想マシン１３のＡＰＩでハードウエアアクセス要求を出力し（ステップＳ１４）、オペレーティングシステム１２のＡＰＩでハードウエアアクセス要求を出力する（ステップＳ１５）。これによって、デバイスドライバ２２がハードウエア１１にアクセスし（ステップＳ１６）デバイスドライバ２２が結果を返す（ステップＳ１７）。この後、アプリケーション１４は、仮想マシン１３のＡＰＩでデバイスドライバ２２をクローズし（ステップＳ１８）、オペレーティングシステム１２のＡＰＩでデバイスドライバ２２をクローズする（ステップＳ１９）という処理が必要である。

【００１１】あるいは、図９に示すように、周辺機器を制御するための共通デバイスドライバ２４を設けておき、アプリケーション１４が共通インターフェイス（Ｉ／Ｆ）２３を介して共通デバイスドライバ２４でハードウエア１１にアクセスする場合もある。

【００１２】しかし、これらの方法では、アプリケーションとは別にアセンブラやＣ言語で（共通）デバイスドライバを記述しなければならないので、デバイスドライバの製作が必要となるために開発効率が悪く、それ以前と生産性は変わらなかった。また、共通インターフェイスを通してハードウエアにアクセスする方法では、細かいハードウエアの制御ができなかった。さらには、ＡＰＩが多重にコールされるので処理速度が遅いという問題があった。

【００１３】また、組込み機器では言語の共通性と同時にアプリケーション１４からハードウエア１１を細かく制御できることが重要である。そのため、最近の取り組みでは、オペレーティングシステムやハードウエアに直接アクセスできるバイパスのＡＰＩを用意して対応している。しかし、仮想マシン１３を実装したシステムでは、割込み等のオペレーティングシステム１２の動作に重要なリソースはオペレーティングシステム１２が一元管理しており、図１０のように仮想マシン１３をバイパスすることは不可能であった。

【００１４】次に、従来の割り込み処理装置では、図１１に示すようにＣＰＵ１６の内部に設けられた割り込みコントローラ２６により、あるいは図１２に示すようにＣＰＵ１６の外部に設けられた割り込みコントローラ２６により、周辺機器からの割り込みを受け付けると共に割り込みソースの状態を監視することによって割り込み処理を実行していた。

【００１５】図１３はＣＰＵ１６の内部に割り込みコントローラ２８を設けた割込み処理装置を表している。外部割り込み要求が許可されると、割り込みソース２７からの割り込みが発生し（Ｃ１）、割り込みコントローラ２８が割り込みを受け付ける。そして、割り込みを受け付けるとオペレーティングシステム１２が割り込みブロックして（Ｃ２）他の割り込みを禁止する。ついで、ＯＳ割り込みハンドラ２９が起動され（Ｃ３）、セマフォによりＪａｖａ仮想マシン割り込みハンドラ３０に起床を通知する（Ｃ４）。Ｊａｖａ仮想マシン割り込みハンドラ３０が起床すれば、Ｊａｖａ仮想マシン１３による割り込み処理が発生する（Ｃ５）。起動したＯＳ割り込みハンドラ２９は、割り込みコントローラ２８と交信することによって割り込み処理の進行を監視する（Ｃ６）。そして、割り込み処理が終了すると、ＯＳ割り込みハンドラ２９が割り込みソース２７の割り込み状態を解除し（Ｃ７）、オペレーティングシステム１２が割り込みブロックを解除すると（Ｃ８）、ＯＳ割り込みハンドラ２９はハンドラから抜け出る（Ｃ９）。

【００１６】このように従来の割り込み処理装置では、割り込み処理はオペレーティングシステムによって一括管理されており、アプリケーションや仮想マシンで制御することはできなかった。このため、上記のようにデバイスドライバによってハードウエアを制御することにより割り込み実行を行わねばならず、アプリケーションの作成に伴ってデバイスドライバを作成する労力が必要であった。

【００１７】

【発明の開示】本発明の目的とするところは、デバイスドライバを用いることなく、アプリケーションから直接ハードウエアアクセス処理を実行できるようにすることにある。

【００１８】本発明にかかる組込み機器用仮想マシンシステムは、アプリケーションと、仮想マシンと、オペレーティングシステムとによって構成され、オペレーティングシステムによって管理される第１のハードウエアアクセス手段と、アプリケーション又は仮想マシンによって管理される第２のハードウエアアクセス手段とを備えたものである。

【００１９】本発明にかかる組込み機器用仮想マシンシステムは、オペレーティングシステムに管理されるハードウエアアクセス手段とは別にアプリケーション又は仮想マシンによって管理されるハードウエアアクセス手段を備えているので、アプリケーション又は仮想マシンによってもハードウエアアクセス処理を実行することができる。従って、アプリケーションや仮想マシンからハードウエアにアクセスするのにデバイスドライバを書く必

要が無く、ハードウエアにアクセスするためのプログラムをアプリケーション内に書くことができる。よって、アプリケーションの開発作業を容易にすることができる。

【００２０】本発明の実施形態においては、前記第２のハードウエアアクセス手段が、前記第１のハードウエアアクセス手段と関連してハードウエアアクセス処理を実行するようになっている。アプリケーション又は仮想マシンによって管理されるハードウエアアクセス手段はオペレーティングシステムによって管理されるハードウエアアクセス手段と関連しているから、オペレーティングシステムがハードウエアへのアクセスを開始した後、オペレーティングシステムから引き継いでアプリケーション又は仮想マシンがハードウエアに継続してアクセスすることが可能になる。よって、アプリケーションや仮想マシンから直接にハードウエアにアクセスできないようなシステムでも、アプリケーションや仮想マシンがハードウエアにアクセスできるようになる。

【００２１】また、本発明の別な実施形態においては、前記第１及び第２のハードウエアアクセス手段は、いずれも割込みコントローラであって、オペレーティングシステムの割込ハンドラと関連しながらアプリケーション又は仮想マシンにより割込み処理を実行できるようになっている。特に、割込み処理の場合には、オペレーティングシステムが割込を受け付けた後、割込みハンドラによって割込み処理をアプリケーション又は仮想マシンに渡すことができる。

【００２２】特に、仮想マシンとしては、Ｊａｖａ仮想マシンが注目されており、組込み機器用としても対応が容易になる。

【００２３】なお、この発明の以上説明した構成要素は、可能な限り組み合わせることができる。

【００２４】

【発明の実施の形態】（第１の実施形態）本発明の一実施形態による組込み機器用仮想マシンシステムを図１４に示す。この組込み機器用仮想マシンシステムにあっては、ハードウエア３１とオペレーティングシステム３２との間に割込み制御装置（ハードウエアアクセス手段）３５が設けられており、オペレーティングシステム３２の上に実装されたＪａｖａ仮想マシン３３の上でアプリケーション３４が実行される。

【００２５】この組み込み機器用仮想マシンシステムにあっては、ハードウエア３１からの割込み要求は、割込み制御装置３５を介してオペレーティングシステム３２に入力される。割込み要求を受け付けると、オペレーティングシステム３２は割込み処理を実行する。アプリケーション３４は、オペレーティングシステム３２が割り込み処理するのに関連して、あるいは協調して割込み制御装置３５にアクセスし、割込み制御装置３５により割り込み処理を実行する。

【００２６】アプリケーション３４が割込み制御装置３５にアクセスする方法としては、仮想マシン３３をバイパスするＡＰＩによってもよく、仮想マシン３３からオペレーティングシステム３２をバイパスしてアクセスしてもよい。

【００２７】しかして、このシステムによれば、オペレーティングシステム３２と協調してアプリケーション３４から直接割込み制御装置３５にアクセスして割込を制御できる。

【００２８】このようにオペレーティングシステム３２とハードウエア３１との間に割込み制御装置３５を設け、アプリケーション３４から割込み制御装置３５を制御できるようにしているので、従来のようにデバイスドライバを書く必要がなく、アプリケーション３４中に割り込み処理のためのプログラムを書くことができ、アプリケーション３４の生産性を向上させることができる。また、従来のように多重にＡＰＩが起動されることもないので、処理速度が向上する。

【００２９】アプリケーション３４がハードウエアにアクセスする方法としては、例えば仮想マシン３３をバイパスして割込み制御装置３５にアクセスしてもよく、仮想マシン３３を経由して割込み制御装置３５にアクセスしても良い。図１５は仮想マシン３３をバイパスするＡＰＩにより割込み制御装置３５を制御する方法を示している。この方法では、図１５のフローに示すように、まず非同期アクセス信号によりアクセス要求があると（ステップＳ２１）、仮想マシン３３をバイパスするＡＰＩで割込み制御装置３５にアクセスするＩ／Ｏオブジェクトを作成し（ステップＳ２２）、このＩ／Ｏオブジェクトを使用してハードウエア３１にアクセスする（ステップＳ２３）。この後、仮想マシン３３をバイパスするＡＰＩで割込み制御装置３５にアクセスするＩ／Ｏオブジェクトを削除する（ステップＳ２４）。

【００３０】また、図１６はハードウエアにアクセスする別な方法を表している。すなわち非同期アクセス信号によりアクセス要求があると（ステップＳ３１）、仮想マシン３３をバイパスするＡＰＩにより割込み制御装置３５にアクセスするＩ／Ｏオブジェクトを作成し（ステップＳ３２）、そのＩ／Ｏオブジェクトを使用してハードウエアアクセス手段（割込み制御装置３５）にアクセス要求する（ステップＳ３３）。ハードウエアアクセス手段はオペレーティングシステム３２の機能を制限し、あるいはその動作を変更し（ステップＳ３４）、Ｉ／Ｏオブジェクトを使用してハードウエア３１にアクセスする（ステップＳ３５）。そして、割り込み処理が終了したら、ハードウエアアクセス手段（割込み制御装置３５）はオペレーティングシステム３２の機能を制限し、あるいはその動作の変更を解除し（ステップＳ３６）、ついで仮想マシン３３をバイパスするＡＰＩでＩ／Ｏオブジェクトを削除する（ステップＳ３７）。

【００３１】図１７はハードウエア３１とオペレーティングシステム３２との間に外部割り込みコントローラ（割込み制御装置）３５を設けて割り込み処理をアプリケーション３４から操作できるようにするための構造を示している。この割り込み処理方法では、オペレーティングシステム３２が外部割り込み要求に対して割り込みを許可する（Ｃ１１）と、割り込みソース３７から割り込みが発生し（Ｃ１２）、外部割込みコントローラ３５が割り込みを受け付け、外部割込みコントローラ３５からＣＰＵ３６内部の割込みコントローラ３８に割り込みが入力される（Ｃ１３）。そして、割り込みを受け付けると、ＣＰＵ３６によってオペレーティングシステム１２が割り込みブロックして（Ｃ１４）他の割り込みを禁止する。ついで、統合割り込みハンドラ（ＯＳレベル）が起動される（Ｃ１５）。統合割り込みハンドラ３９が追加割り込みブロック機能の外部割込みを不許可にし（Ｃ１６）、セマフォによりＪａｖａ仮想マシン割込みハンドラ４０に起床を通知し（Ｃ１７）、統合割込みハンドラ３９が割り込み処理のループから抜け（Ｃ１８）、オペレーティングシステム３２が割込みコントローラ３８における割込みブロックを解除する（Ｃ１９）。一方、ｊａｖａ仮想マシン割込みハンドラ４０は起床してＪａｖａ仮想マシン３３による割り込み処理を行い（Ｃ２０）、外部割込みコントローラ３５（もしくは、割込みソース３７）と交信することによって割込処理を監視する（Ｃ２１）。Ｊａｖａ仮想マシン３３による割込み処理が終了すると、Ｊａｖａ仮想マシン割り込みハンドラ４０が割込み処理を終了し（Ｃ２２）、統合割込みハンドラ３９が割込みコントローラ３８において外部割り込みを許可にする（Ｃ２３）。

【００３２】なお、図１７では、割り込みコントローラはＣＰＵ内部に設けられているが、外部割り込みコントローラと共に割り込みコントローラもＣＰＵの外部に設けられていてもよい。

【００３３】

【発明の効果】本発明の組込み機器用仮想マシンシステムによれば、ハードウエアアクセス手段が二重化されており、アプリケーション等によって管理されている第２のハードウエアアクセス手段は、オペレーティングシステム側のハードウエアアクセス手段と関連して動作するので、アプリケーションから直接ハードウエアアクセス処理を実行できるようになり、アプリケーションの生産性向上と処理速度の向上を図ることができる。

【図面の簡単な説明】

【図１】従来の一般的なコンピュータシステムの構成を示す図である。

【図２】仮想マシンを備えた従来のコンピュータシステムの構成を示す図である。

【図３】マイクロプロセッサと周辺機器との関係を示す図である。

【図４】図１のコンピュータシステムにおいて、アプリケーションからハードウエアを直接制御する様子を示す図である。

【図５】同上のコンピュータシステムにおいて、ハードウエアにアクセスする手順を説明するフロー図である。

【図６】仮想マシンを備えたコンピュータシステムにおいて、アプリケーションからオペレーティングシステム又はハードウエアへアクセスすることができない様子を表した図である。

【図７】デバイスドライバと共通インターフェイスを用いてアプリケーションからハードウエアへアクセスできるようにしたコンピュータシステムを示す図である。

【図８】同上の方法によりアプリケーションからハードウエアへアクセスする手順を説明する図である。

【図９】共通デバイスドライバと共通インターフェイスを用いてアプリケーションからハードウエアへアクセスできるようにしたコンピュータシステムを示す図である。

【図１０】仮想マシンをバイパスするＡＰＩを用いてアプリケーションからハードウエアへアクセスできるようにしたコンピュータシステムを示す図である。

【図１１】マイクロプロセッサの内部に割込みコントローラを設けた従来のシステムを示す図である。

【図１２】マイクロプロセッサの外部に割込みコントローラを設けた従来のシステムを示す図である。

【図１３】従来の割込み処理装置の構成を示すブロック図である。

【図１４】本発明の一実施形態による組込み機器用仮想マシンシステムの構成を示す概略図である。

【図１５】同上の組込み機器用仮想マシンにおいて、ハードウエアにアクセスする方法を説明するフロー図である。

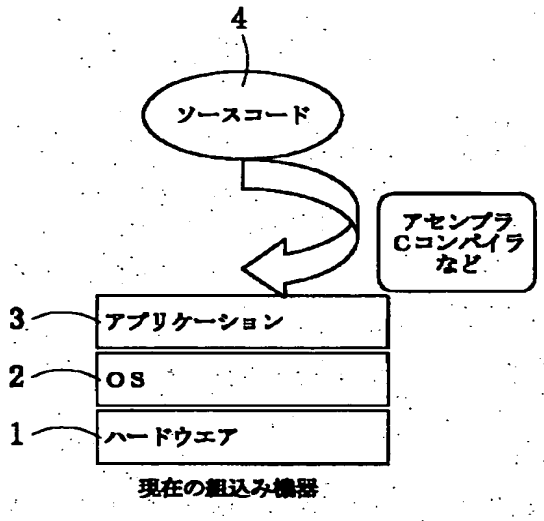
【図１６】図１４の組込み機器用仮想システムにおいて、ハードウエアにアクセスする別な方法を説明するフロー図である。

【図１７】本発明にかかる割込み処理装置の構成を示す図である。

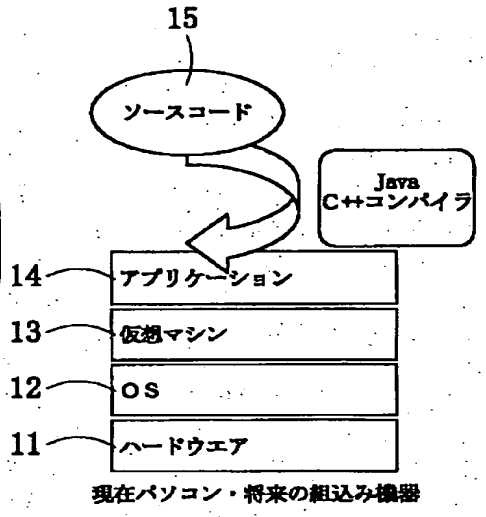
【符号の説明】

３１　ハードウエア
３２　オペレーティングシステム
３３　仮想マシン
３４　アプリケーション
３５　制御装置

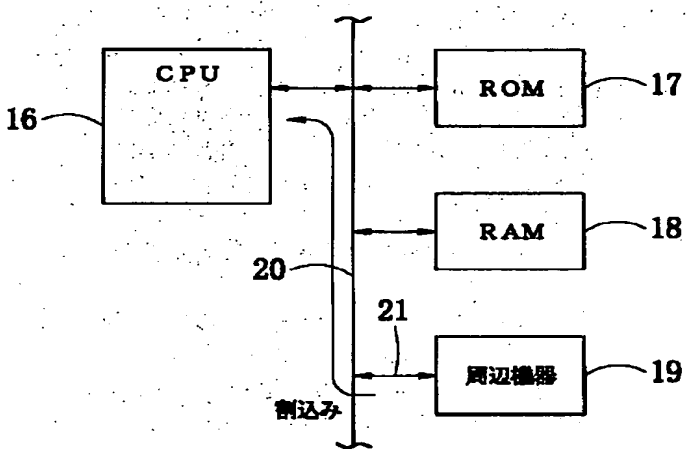
【図１】



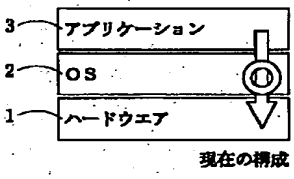
【図２】



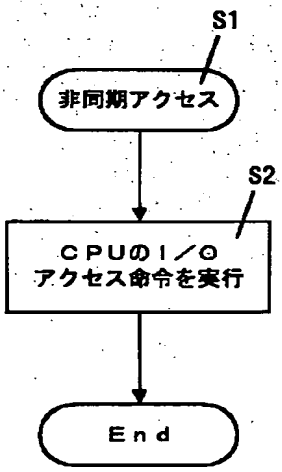
【図３】



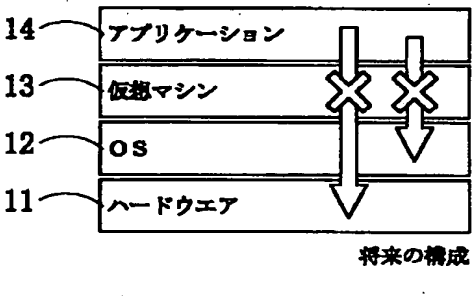
【図４】



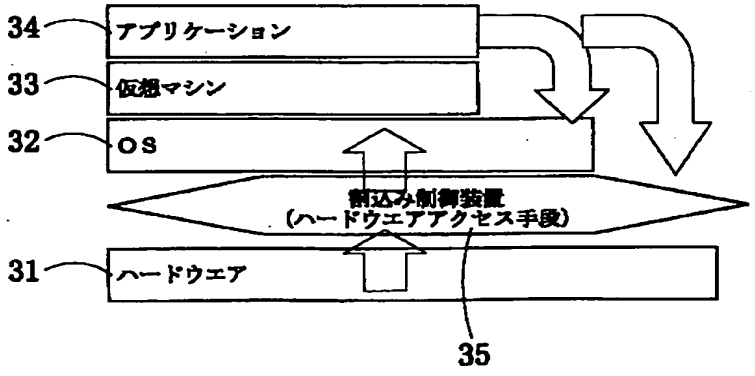
【図５】



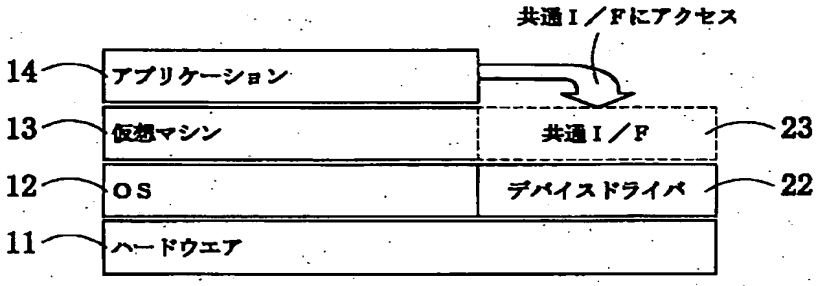
【図６】



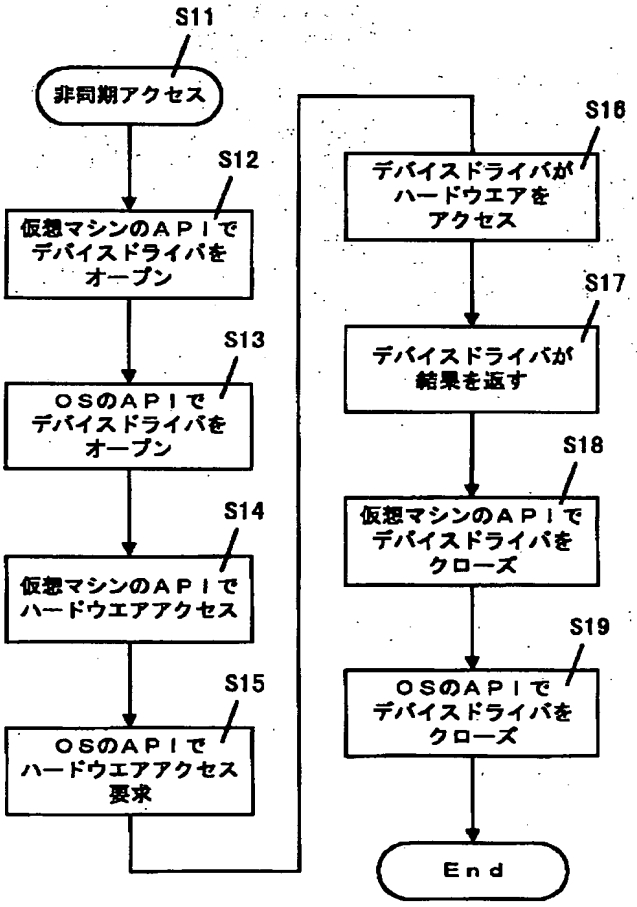
【図１４】



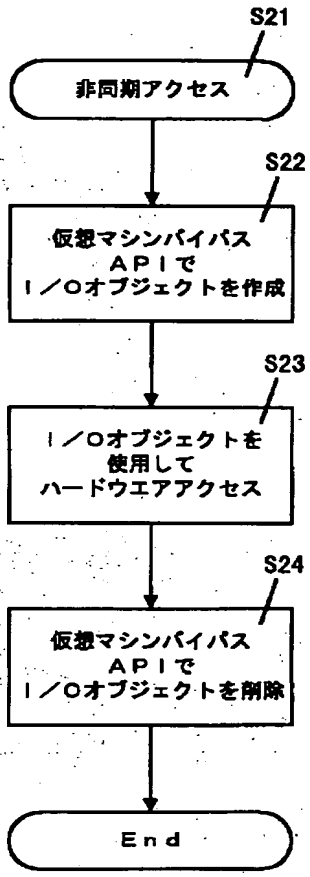
【図７】


共通Ｉ／Ｆにアクセス
14
アプリケーション
13
仮想マシン
共通Ｉ／Ｆ
23
12
ＯＳ
デバイスドライバ
22
11
ハードウエア

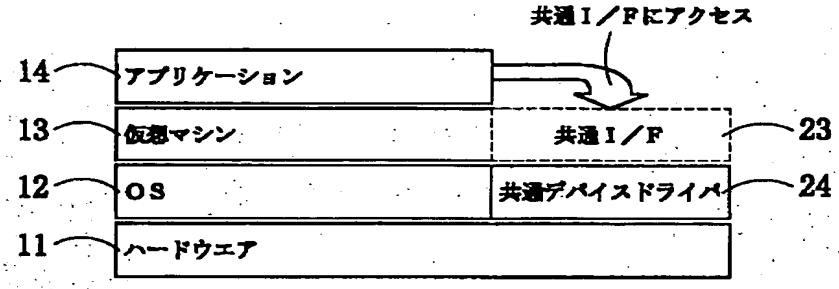
【図８】


S11
非同期アクセス
S12
仮想マシンのＡＰＩで
デバイスドライバを
オープン
S13
ＯＳのＡＰＩで
デバイスドライバを
オープン
S14
仮想マシンのＡＰＩで
ハードウエアアクセス
S15
ＯＳのＡＰＩで
ハードウエアアクセス
要求
S16
デバイスドライバが
ハードウエアを
アクセス
S17
デバイスドライバが
結果を返す
S18
仮想マシンのＡＰＩで
デバイスドライバを
クローズ
S19
ＯＳのＡＰＩで
デバイスドライバを
クローズ
End

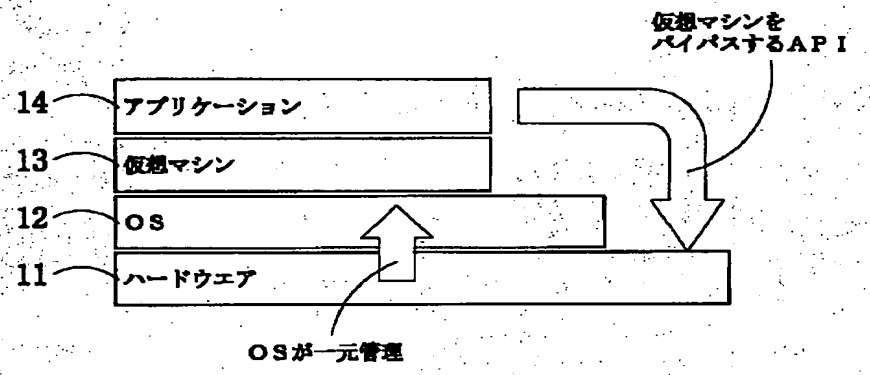
【図１５】


S21
非同期アクセス
S22
仮想マシンバイパス
ＡＰＩで
Ｉ／Ｏオブジェクトを作成
S23
Ｉ／Ｏオブジェクトを
使用して
ハードウエアアクセス
S24
仮想マシンバイパス
ＡＰＩで
Ｉ／Ｏオブジェクトを削除
End

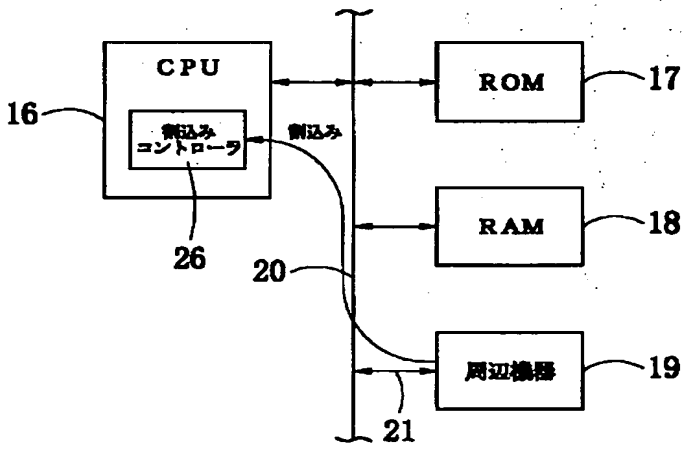
【図９】


共通Ｉ／Ｆにアクセス
14
アプリケーション
13
仮想マシン
共通Ｉ／Ｆ
23
12
ＯＳ
共通デバイスドライバ
24
11
ハードウエア

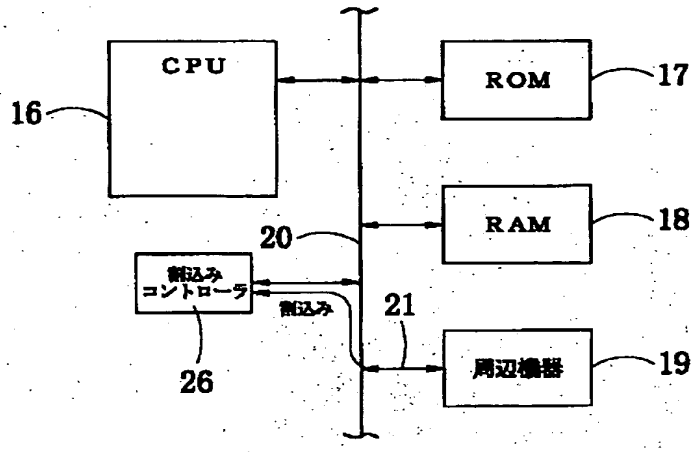
【図１０】


仮想マシンを
バイパスするＡＰＩ
14
アプリケーション
13
仮想マシン
12
ＯＳ
11
ハードウエア
ＯＳが一元管理

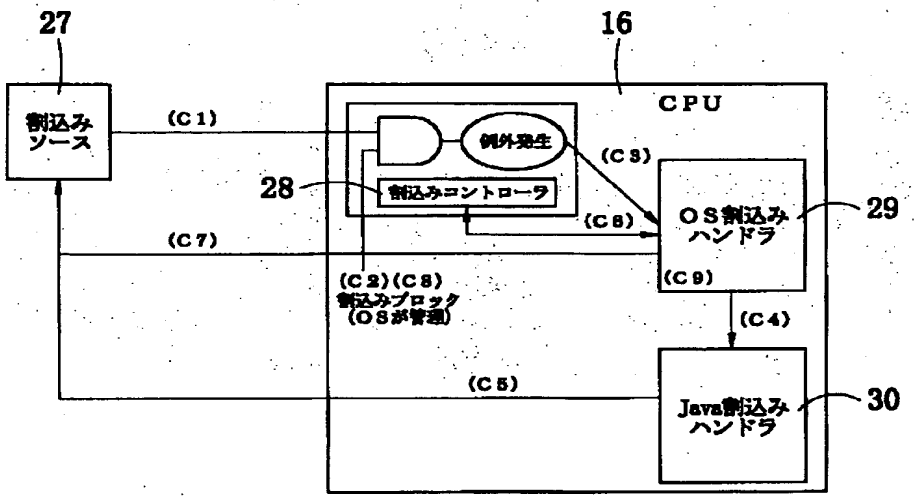
【図１１】


CPU
16
割込み
コントローラ
26
割込み
ROM
17
RAM
18
20
周辺機器
19
21

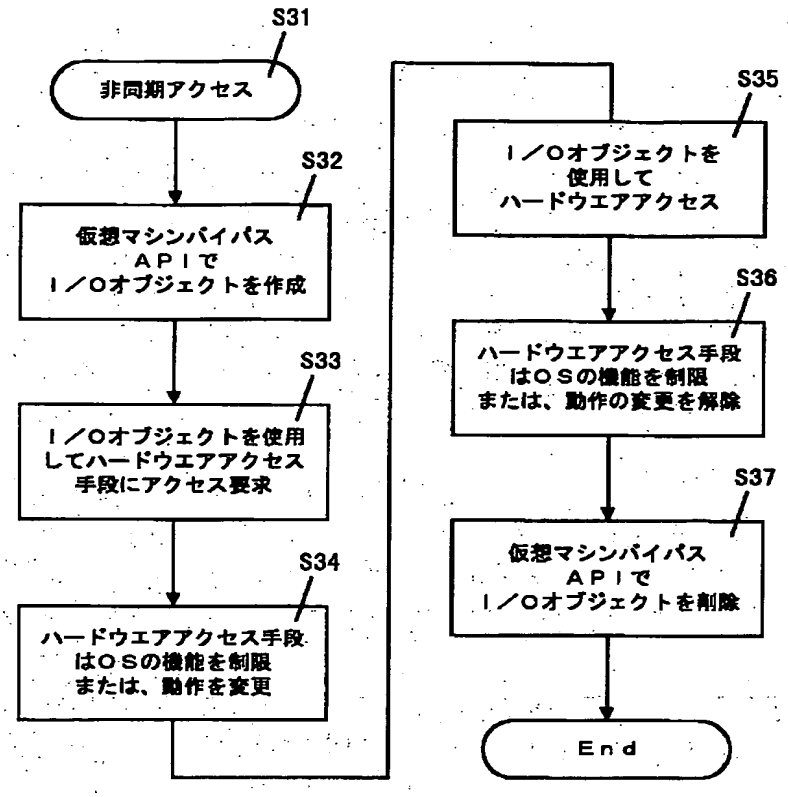
【図１２】


CPU
16
ROM
17
RAM
18
20
割込み
コントローラ
割込み
21
周辺機器
19
26

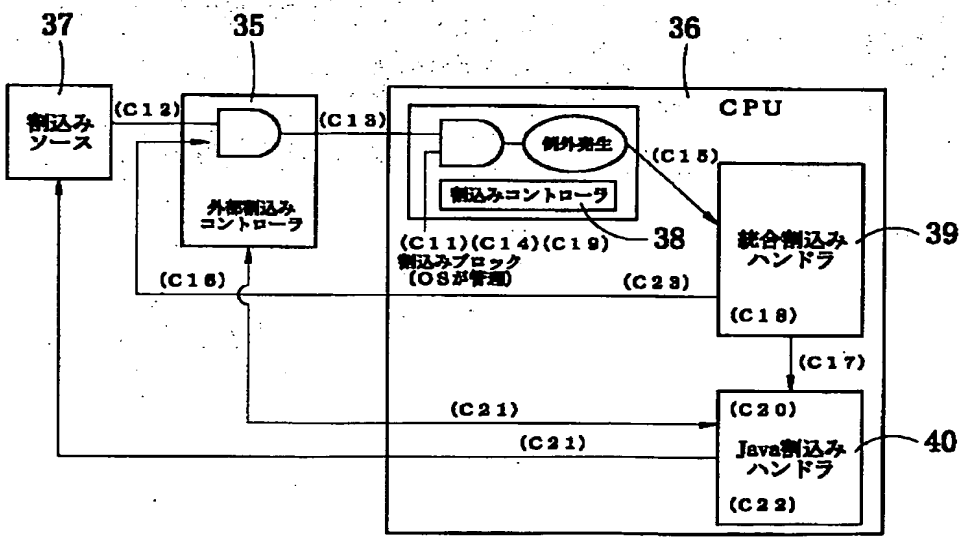
【図１３】


27
16
CPU
割込み
ソース
(C1)
例外発生
(C3)
28
割込みコントローラ
(C8)
OS割込み
ハンドラ
29
(C7)
(C2)(C8)
割込みブロック
(OSが管理)
(C9)
(C4)
(C5)
Java割込み
ハンドラ
30

【図１６】


S31
非同期アクセス
S32
仮想マシンパイパス
ＡＰＩで
Ｉ／Ｏオブジェクトを作成
S33
Ｉ／Ｏオブジェクトを使用
してハードウエアアクセス
手段にアクセス要求
S34
ハードウエアアクセス手段
はＯＳの機能を制限
または、動作を変更
S35
Ｉ／Ｏオブジェクトを
使用して
ハードウエアアクセス
S36
ハードウエアアクセス手段
はＯＳの機能を制限
または、動作の変更を解除
S37
仮想マシンパイパス
ＡＰＩで
Ｉ／Ｏオブジェクトを削除
End

【図１７】


37
35
36
CPU
割込み
ソース
(C12)
(C13)
外部割込み
コントローラ
例外発生
(C15)
割込みコントローラ
(C11)(C14)(C19)
38
割込みブロック
(OSが管理)
(C16)
(C23)
統合割込み
ハンドラ
39
(C18)
(C17)
(C21)
(C20)
(C21)
Java割込み
ハンドラ
40
(C22)

フロントページの続き

(72)発明者 門脇 正規
京都府京都市右京区花園土堂町10番地 オムロン株式会社内
(72)発明者 村井 謙一
京都府京都市右京区花園土堂町10番地 オムロン株式会社内
(72)発明者 齋木 秀明
京都府京都市下京区西洞院木津屋橋通東入ル オムロンソフトウエア株式会社内

Ｆターム(参考) 5B014 EA01 EB03 FA09 GD02 GD05 GD22 GD32 HC12
5B098 AA05 BB05 BB18 GA02 GD07 GD14